Conhecendo o MQTT

Por que o MQTT é um dos melhores protocolos de rede para a Internet das Coisas?

Michael Yuan (michael@michaelyuan.com)
Independent consultant

03/Out/2017
(Primeira publicação 04/Out/2017)

Este artigo oferece uma introdução técnica sobre o protocolo MQTT. Você aprenderá o que é o MQTT, por que ele é adequado para aplicativos de IoT e como começar a desenvolver aplicativos que usam o MQTT.

Para os dispositivos de Internet das Coisas (IoT), a conexão com a Internet é um requisito. A conexão com a Internet permite que os dispositivos trabalhem entre si e com serviços de backend. O protocolo de rede subjacente da Internet é o TCP/IP. Desenvolvido com base na pilha TCP/IP, o MQTT (Message Queue Telemetry Transport) tornou-se o padrão para comunicações de IoT.

O MQTT foi inventado e desenvolvido inicialmente pela IBM no final dos anos 90. Sua aplicação original era vincular sensores em pipelines de petróleo a satélites. Como seu nome sugere, ele é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS, com suporte nas linguagens de programação populares, usando diversas implementações de software livre.

Por que o MQTT

O MQTT é um protocolo de rede leve e flexível que oferece o equilíbrio ideal para os desenvolvedores de IoT:

- O protocolo leve permite a implementação em hardware de dispositivo altamente restringido e em redes de largura da banda limitada e de alta latência.
- Sua flexibilidade possibilita o suporte a diversos cenários de aplicativo para dispositivos e serviços de IoT.

Para entender por que o MQTT é tão adequado para desenvolvedores de IoT, vamos analisar por que outros protocolos de rede populares falharam em IoT.

Por que não usar algum dos outros inúmeros protocolos de rede

A maioria dos desenvolvedores já se acostumou aos serviços da Web HTTP. Então, por que não conectar os dispositivos de IoT aos serviços da web? O dispositivo poderia enviar seus dados como uma solicitação de HTTP e receber atualizações do sistema como uma resposta de HTTP. Esse padrão de solicitação e resposta tem algumas limitações graves:

- O HTTP é um protocolo síncrono. O cliente espera que o servidor responda. Os navegadores da web têm esse requisito, mas o custo é a baixa escalabilidade. No mundo da IoT, a comunicação síncrona tem sido um problema devido ao grande número de dispositivos e à rede, muitas vezes não confiável e de alta latência. Um protocolo de mensagem assíncrono é muito mais adequado para aplicativos de IoT. Os sensores podem enviar leituras e permitir que a rede descubra o caminho e a sincronização ideais para entregar aos dispositivos e serviços de destino.
- HTTP é unidirecional. O cliente precisa iniciar a conexão. Em um aplicativo de IoT, os dispositivos e sensores geralmente são clientes, o que significa que eles não podem receber comandos da rede passivamente.
- HTTP é um protocolo de um para um. O cliente faz uma solicitação e o servidor responde. É difícil e caro transmitir uma mensagem a todos os dispositivos na rede, o que é um caso de uso comum em aplicativos de IoT.
- HTTP é um protocolo pesado com muitos cabeçalhos e regras. Ele não é adequado para redes restringidas.

Pelos motivos citados acima, a maioria dos sistemas escaláveis de alto desempenho usam um barramento do sistema de mensagens assíncrono, em vez de serviços da web, para trocas de dados internas. Na verdade, o protocolo de sistema de mensagens mais popular que é usado em sistemas de middleware corporativos é chamado AMQP (Advanced Message Queuing Protocol). No entanto, no ambiente de alto desempenho, a capacidade de computação e a latência da rede geralmente não são uma preocupação. O AMQP foi criado para assegurar a confiabilidade e a interoperabilidade em aplicativos corporativos. Ele possui um rico conjunto de recursos, mas não é adequado para aplicativos de IoT com restrição de recursos.

Além do AMQP, existem outros protocolos populares de sistema de mensagens. Por exemplo, o XMPP (Extensible Messaging and Presence Protocol) é um protocolo de mensagem instantânea (IM) ponto a ponto. Ele é pesado em recursos com suporte para casos de uso de IM, como presença e anexos de mídia. Em comparação com o MQTT, ele requer muito mais recursos no dispositivo e na rede.

Então, como o MQTT consegue ser tão leve e flexível? Um importante recurso do protocolo MQTT é o modelo de publicação e assinatura. Como em todos os protocolos de sistema de mensagens, ele desacopla o publicador e o consumidor de dados.

O modelo de publicação e assinatura

O protocolo MQTT define dois tipos de entidades na rede: um message broker e inúmeros clientes. O broker é um servidor que recebe todas as mensagens dos clientes e, em seguida, roteia essas mensagens para os clientes de destino relevantes. Um cliente é qualquer coisa que possa interagir com o broker e receber mensagens. Um cliente pode ser um sensor de IoT em campo ou um aplicativo em um data center que processa dados de IoT.

1. O cliente conecta-se ao broker. Ele pode assinar qualquer "tópico" de mensagem no broker. Essa conexão pode ser uma conexão TCP/IP simples ou uma conexão TLS criptografada para mensagens sensíveis.
2. O cliente publica as mensagens em um tópico, enviando a mensagem e o tópico ao broker.
3. Em seguida, o broker encaminha a mensagem a todos os clientes que assinam esse tópico.

Como as mensagens do MQTT são organizadas por tópicos, o desenvolvedor de aplicativos tem a flexibilidade de especificar que determinados clientes somente podem interagir com determinadas mensagens. Por exemplo, os sensores publicarão suas leituras no tópico "sensor_data" e assinarão o tópico "config_change". Os aplicativos de processamento de dados que salvam os dados do sensor em um banco de dados de backend assinarão o tópico "sensor_data". Um aplicativo de console administrativo poderia receber comandos do administrador do sistema para ajustar as configurações dos sensores, como a sensibilidade e a frequência de amostragem, e publicar essas mudanças no tópico "config_change". (Consulte Figura 1.)

Figura 1. O modelo de publicação e assinatura do MQTT para sensores de IoT

Ao mesmo tempo, o MQTT é leve. Ele tem um cabeçalho simples para especificar o tipo de mensagem, um tópico baseado em texto e, em seguida, uma carga útil binária arbitrária. O

aplicativo pode usar qualquer formato de dados para a carga útil como JSON, XML, binário criptografado ou Base64, desde que os clientes de destino possam analisar a carga útil.

Introdução ao desenvolvimento com MQTT

A ferramenta mais fácil para começar o desenvolvimento com MQTT é o módulo Python mosquitto, que faz parte do projeto Eclipse Paho e fornece SDKs e bibliotecas do MQTT em várias linguagens de programação. Ele contém um broker do MQTT que pode ser executado no computador local e ferramentas de linha de comandos que podem interagir com o broker usando mensagens. É possível fazer download e instalar o módulo mosquitto no website do mosquitto.

O comando mosquitto executa o broker do MQTT no computador local. Opcionalmente, é possível usar a opção -d para executá-lo em segundo plano.

```
$ mosquitto -d
```

Em seguida, em outra janela do terminal, é possível usar o comando mosquitto_sub para conectar-se ao broker local e assinar um tópico. Após a execução do comando, ele vai esperar e imprimir todas as mensagens que receber da assinatura, à medida que elas forem chegando.

```
$ mosquitto_sub -t "dw/demo"
```

Em uma outra janela do terminal, é possível usar o comando mosquitto_pub para conectar-se ao broker local e, em seguida, publicar uma mensagem em um tópico.

```
$ mosquitto_pub -t "dw/demo" -m "hello world!"
```

Agora, o terminal que executa o mosquitto_sub deverá exibir "hello world!" na tela. Você acabou de enviar e receber uma mensagem usando um broker do MQTT!

É claro que em um sistema de produção, não é possível usar um computador local como o broker. Nesse caso, é possível usar o Serviço da plataforma Internet das Coisas do IBM Bluemix, que é um serviço sob demanda e confiável que funciona como um broker do MQTT. (Leia mais sobre como esse serviço do Bluemix se integra e usa o MQTT como seu protocolo para comunicar-se com dispositivos e aplicativos na documentação do serviço.)

O nó Serviço da plataforma Internet das Coisas do IBM Bluemix funciona da seguinte forma.

- No console do Bluemix, é possível criar uma instância do serviço da plataforma Internet das Coisas sob demanda.
- Em seguida, é possível incluir dispositivos que possam conectar a instância de serviço usando o MQTT. Cada dispositivo terá um ID e um nome. Somente os dispositivos listados podem acessar o serviço, e o painel da plataforma Watson IoT relatará informações sobre tráfego e uso desses serviços, nos respectivos dispositivos.
- Para cada cliente do dispositivo, o Bluemix designará um nome do host, um nome de usuário e uma senha para a conexão com a instância de serviço (o broker do MQTT). (No Bluemix, o

nome de usuário sempre é use-token-auth e a senha é o token que é mostrado na Figura 2 para cada dispositivo conectado.)

Figura 2. Criando uma instância de serviço da plataforma Internet das Coisas no IBM Bluemix

Ao usar um broker remoto do MQTT, será necessário passar pela aprovação das credenciais de autenticação e de nome do host do broker para os comandos mosquitto_sub e mosquitto_pub. Por exemplo, o comando a seguir assina o tópico de demonstração no nosso serviço da plataforma Internet das Coisas com o nome de usuário e senha fornecidos pelo Bluemix:

```
$ mosquitto_sub -t "demo" -h host.iotp.mqtt.bluemix.com -u username -P password
```

Para conhecer mais opções de como usar as ferramentas do mosquitto e também de como usar a API do mosquitto para criar seus próprios aplicativos clientes do MQTT, consulte a documentação no website do mosquitto.

Agora que você já tem as ferramentas necessárias, vamos nos aprofundar no protocolo MQTT.

Entendendo o protocolo MQTT

O MQTT é um protocolo de ligação que especifica como os bytes de dados são organizados e transmitidos pela rede TCP/IP. Mas por motivos práticos, os desenvolvedores não precisam entender o protocolo de ligação. Basta saber que cada mensagem tem uma carga útil de comando e dados. O comando define o tipo de mensagem (por exemplo, uma mensagem CONNECT ou uma mensagem SUBSCRIBE). Todas as bibliotecas e ferramentas do MQTT

oferecem maneiras simples de manipular essas mensagens diretamente e podem preencher automaticamente alguns campos necessários, como os IDs da mensagem e do cliente.

Primeiro, o cliente conecta-se ao broker enviando uma mensagem CONNECT. A mensagem CONNECT pede para estabelecer uma conexão do cliente com o broker. A mensagem CONNECT tem os parâmetros de conteúdo a seguir.

Tablela 1. Parâmetros da mensagem CONNECT

Parâmetro	Descrição
cleanSession	Esta sinalização especifica se a conexão é persistente ou não. Uma sessão persistente armazena todas as assinaturas e as mensagens possivelmente perdidas (dependendo do QoS) no broker. (Consulte Tablela 3 para obter uma descrição do QoS).
username	As credenciais de autenticação e autorização do broker.
password	As credenciais de autenticação e autorização do broker.
lastWillTopic	Quando a conexão for encerrada inesperadamente, o broker publicará automaticamente uma mensagem de "último desejo" em um tópico.
lastWillQos	O QoS da mensagem de "último desejo". (Consulte Tablela 3 para obter uma descrição do QoS).
lastWillMessage	A própria mensagem de "último desejo".
keepAlive	Este é o intervalo de tempo em que o cliente precisa efetuar ping no broker para manter a conexão ativa.

O cliente receberá uma mensagem CONNACK do broker. A mensagem CONNACK tem os parâmetros de conteúdo a seguir.

Tablela 2. Parâmetros da mensagem CONNACK

Parâmetro	Descrição
sessionPresent	Indica se a conexão já tem uma sessão persistente. Ou seja, a conexão já tem tópicos assinados e receberá a entrega de mensagens ausentes.
returnCode	0 indica sucesso. Outros valores identificam a causa da falha.

Depois que uma conexão for estabelecida, o cliente poderá enviar uma ou mais mensagens SUBSCRIBE ao broker para indicar que ele receberá mensagens do broker de determinados tópicos. A mensagem pode ter uma ou várias repetições dos parâmetros a seguir.

Tablela 3. Parâmetros da mensagem SUBSCRIBE

Parâmetro	Descrição
qos	A sinalização qos (qualidade de serviço ou QoS) indica com que consistência as mensagens neste tópico precisam ser entregues aos clientes. • Valor 0: não confiável, a mensagem é entregue no máximo uma única vez, se o cliente estiver indisponível no momento, ele perderá a mensagem. • Valor 1: a mensagem deve ser entregue pelo menos uma vez.

	• Valor 2: a mensagem deve ser entregue exatamente uma vez.
tópico	Um tópico para assinar. Um tópico pode ter vários níveis separados pelo caractere barra. Por exemplo, "dw/demo" e "ibm/bluemix/mqtt" são tópicos válidos.

Depois que o cliente tiver assinado um tópico com sucesso, o broker retornará uma mensagem SUBACK com um ou mais parâmetros returnCode.

Tablela 4. Parâmetros da mensagem SUBACK

Parâmetro	Descrição
returnCode	Existe um código de retorno para cada um dos tópicos no comando SUBSCRIBE. Os valores de retorno são os seguintes. • Valores 0 a 2: sucesso como nível de QoS correspondente. (Consulte Tablela 3 para saber mais sobre QoS.) • Valor 128: falha.

Correspondendo à mensagem SUBSCRIBE, o cliente também poderá UNSUBSCRIBE (cancelar a assinatura) de um tópico ou de vários tópicos.

Tablela 5. Parâmetros da mensagem UNSUBSCRIBE

Parâmetro	Descrição
tópico	Este parâmetro pode se repetir para vários tópicos.

O cliente pode enviar mensagens PUBLISH ao broker. A mensagem contém um tópico e uma carga útil de dados. Em seguida, o broker encaminha a mensagem a todos os clientes que assinam esse tópico.

Tablela 6. Parâmetros da mensagem PUBLISH

Parâmetro	Descrição
topicName	O tópico no qual a mensagem é publicada.
qos	O nível de qualidade de serviço da entrega da mensagem. (Consulte Tablela 3 para obter uma descrição do QoS).
retainFlag	Esta sinalização indica se o broker reterá a mensagem como a última mensagem conhecida deste tópica.
carga útil	Os dados reais na mensagem. Pode ser uma sequência de texto ou um blob binário de dados.

Dicas e soluções alternativas

O poder do MQTT é a simplicidade. Não há restrições quanto ao tipo de tópico ou de carga útil de mensagem que se pode usar. Isso permite alguns casos de uso interessantes. Por exemplo, considere estas questões:

Como executar mensagens de um para um com o MQTT? Ambas as partes podem concordar em usar um tópico que seja exclusivo para elas. Por exemplo, o nome do tópico pode conter os IDs dos dois clientes para garantir sua exclusividade.

Como um cliente transmite seu status de presença? O sistema pode concordar com uma convenção de nomenclatura para tópicos de "presença". Por exemplo, o tópico "presence/client-id" pode conter as informações de presença do cliente. O cliente define a mensagem como true quando se conecta e false quando se desconecta. O cliente também pode configurar uma mensagem de last will como false para ser definida quando a conexão for encerrada. A mensagem pode ser retida pelo broker para que novos clientes possam ler o tópico e descobrir o status de presença.

Como proteger as comunicações? A conexão do cliente com o broker pode ser uma conexão TLS criptografada para proteger os dados em trânsito. Além disso, como o protocolo MQTT não impõe restrições quanto ao formato de dados da carga útil, o sistema pode concordar com um método de criptografia e um mecanismo de atualização de chave. Depois disso, todo o conteúdo na carga útil pode ser dados binários criptografados das mensagens JSON ou XML reais.

Conclusão

Neste artigo eu ofereci uma introdução técnica ao protocolo MQTT. Você aprendeu o que é o MQTT, por que ele é adequado para aplicativos de IoT e como começar a desenvolver aplicativos que usam o MQTT.

Em um de meus próximos artigos, vou mostrar como desenvolver uma solução de sensor de IoT completa com serviços de MQTT de backend e usando uma placa NodeMCU.

Recursos

Temas relacionados: O website oficial do MQTT MQTT v3.1.1 é um padrão OASIS oficial O projeto Eclipse Paho inclui implementações do MQTT de software livre populares O projeto mosquitto, um broker e uma biblioteca do cliente do MQTT Python de software livre Documentação do MQTT para o serviço da plataforma Internet das Coisas

Sobre o autor

Michael Yuan

Dr. Michael J. Yuan is a recognized expert in mobile end-to-end solutions with J2ME, J2EE, and .Net technologies, having authored more than 30 technical and academic articles on this subject. He is the author of two books, *Enterprise J2ME* (Prentice Hall, 2003) and *Developing Scalable Series 40 Applications* (Addison-Wesley and Nokia, 2004).